

Spying Browser Extensions: Analysis and Detection

Anupama Aggarwal
IIIT-Delhi, India
anupamaa@iiitd.ac.in

Bimal Viswanath
UC Santa Barbara
viswanath@cs.ucsb.edu

Saravana Kumar
CEG, Guindy, India
savkumar90@gmail.com

Liang Zhang
Northeastern University
liang@ccs.neu.edu

Ayush Shah
IIIT-Delhi, India
ayush13027@iiitd.ac.in

Ponnuramgam
Kumaraguru
IIIT-Delhi, India
pk@iiitd.ac.in

ABSTRACT

Several studies have been conducted on understanding third-party user tracking on the web. However, web trackers can only track users on sites where they are embedded by the publisher, thus obtaining a fragmented view of a user’s online footprint. In this work, we investigate a different form of user tracking, where extensions enabled on a browser can capture the complete browsing behavior of a user and communicate the collected sensitive information to a remote server (untrusted by the user). We conduct the first large-scale empirical study of 218 *spying browser extensions* on the Chrome Web Store. We observe that these extensions steal a variety of sensitive user information, such as the complete browsing history of the users (e.g., the sequence of web traversals), online social network (OSN) access tokens, and IP address and geolocation of users. We present an in-depth analysis of the spying behavior of these extensions. Finally, we investigate the potential for automatically detecting spying extensions by applying machine learning schemes using a comprehensive set of features capturing various client-side and network behavior. Our findings highlight the importance of detecting and limiting user behavior tracking by browser extensions.

1. INTRODUCTION

Most sites on the web are increasingly relying on behavioral advertising to sustain their services. Online advertising in the US hit a record 16 Billion USD of revenue in 2016 [6]. At its core, the largest online ad platforms are based on wide-spread sophisticated user behavior tracking. This has received immense attention in both industry and academia. Several studies have been conducted to understand *third-party user tracking on the web*, where an entity (the third-party tracker), embedded by the site (first party) the user visits, is able to log the presence and other activities of the user on that site [9, 30].

In this work, we take a step towards understanding a significantly different type of user tracking—user *spying* (or spying for short) by *browser extensions*. Spying violates user privacy by collecting sensitive personal information without consensus from the user. This information can later be used towards targeted attacks, or can be traded in underground markets. Unfortunately, user tracking, which started as a means to enable better personalized advertising, has reached to a state where even privileged software running on a user’s browser is capturing sensitive personal information and sending it to remote servers (untrusted by the user).

But one might still wonder why spying by browser extensions deserves special attention, when studies have mainly focused on third-party tracking on the web. First, it should be noted that today, the browser is the most widely used software and serves as a window to the web. In fact, platforms such as ChromeOS further reinforce the concept that “your browser is your operating system” [28]. Second, most popular browsers (e.g., Chrome, Safari, Firefox) provide an *extension platform* that enables third-party software to reside on the browser to improve its core functionality. For example, the Chrome Web Store has a very popular extension platform hosting over 43,000 extensions, with some individual extensions having over 9 million users. These extensions enable a wide range of additional useful functionality such as finding better deals in online marketplaces [18], and checking spelling and grammar [15]. Unfortunately, not all extensions are benign. Extensions can acquire a significant amount of (unwanted) privileges that would allow them to capture a “complete” online browsing footprint of a user, ranging from capturing website visits, mouse movements/clicks, and keypresses as well as access to various browser storage mechanisms. Lastly, it is important to note that unlike extensions, third-party web trackers can only track users on sites where they are embedded by the publisher, and thus obtain a “limited” or fragmented view of a user’s entire browsing behavior. Extensions enabled on a browser can capture the complete browsing behavior of a user across any site and potentially across multiple devices of the user (as long as the same browser and extension is used across devices).

Prior work on malicious browser extensions has mostly investigated ad manipulation [35], and online social network (OSN) abuse [19] in detail. In this work, we present the first large-scale empirical study of 218 spying extensions on the Chrome Web Store [14]. As of the time of writing, 214 out of 218 extensions were still alive on the Chrome Web Store and could be installed by Chrome users. We observe that these spying extensions “stealthily” collect and send a variety of sensitive private information, such as the complete browsing history of the users, including the sequence of web traversals, certain OSN access tokens as well as IP address and geolocation of users to untrusted remote servers. From our investigation, we also observe that none of these extensions would need to collect such information for its functionality. It is clear that leakage of these types of information (e.g., browsing history) is a huge violation of user privacy. For example, a browsing history can contain URLs referencing private documents (e.g., Google docs) or the parameters of the URL can reveal various privacy sensitive activities

performed by the user on a site (e.g., online e-commerce transactions, password-based authentication). Further, the remote entity which is collecting the personal data can potentially sell it for profit or even enable a targeted attack (e.g., spear phishing attack) against the user by leveraging the personal information.

The first challenge to conduct this study is to identify a large sample of spying extensions. Using expert manual investigation and semi-automated heuristic based techniques, we analyze 43k extensions and identify a dataset of 218 spying extensions (described in Section 3). Next, we present an in-depth analysis of these spying extensions (Section 4) to understand the different types of sensitive information being stolen as well as their spying behavior. Our analysis shows that users are mostly unaware of the spying behavior of these extensions. In fact, the 218 spying extensions receive over 2.4 million cumulative installs in the Chrome store.

Finally, we investigate the potential for automatically detecting these spying extensions using machine learning schemes (Section 5). We develop a comprehensive set of features for the classification problem, and observe that features based on communication end points with the browser (Chrome API) provide the highest accuracy. Also, among all the classifiers we applied, we find that Neural Networks provide the highest precision and recall. Overall, our work highlights the importance and feasibility of detecting and limiting spying within the browser.

2. EXTENSION ARCHITECTURE

Browser extensions are software programs that can extend the functionality of web browsers. In this section, we briefly overview the Chrome extension architecture. Although we focus on Chrome extension, most of the concept apply to other browsers as well, for example, Firefox with WebExtensions¹.

A chrome extension has two components: a manifest file and web objects (e.g., HTML, CSS, and JavaScript files). A manifest file is for an extension to declare its required permissions (e.g., access permission for storage, user cookies, and/or external URLs) and security contexts (e.g., sandbox, Content Security Policy, and extension’s externally connectivities). Most manifest files specify a background page—a HTML page or a JavaScript file—as the main entry point of the extension. All dependent web objects, including ones required by the background page, are packed as resources in the extension.

In Chrome, a extension has two execution models: a long-running model called background pages and a short page-based model called content scripts. The extension can have scripts running in both models at the same time. The background page is an isolated sandbox environment for the extension to perform tasks lasting across the whole browser life cycle. For example, extension developers use background pages to monitor browser-wide events, such as tabbing and networking. Background pages are also used for keeping states and sharing information for content scripts.

Contrary to the long-lasting background pages, content scripts run in the context of the visiting web pages (i.e., they has the same life cycle as the web pages). Each page has its own content scripts instance, while a background page is

a singleton shared among all web pages. Following declarations in the manifest file, Chrome injects content scripts into the targeted web pages to which the user navigate, as if they were loaded by the web site. To prevent unsafe cross-site scripting, the extension is set to the security context defined in the manifest file.

Besides the two models have different life cycles, they access to two sets of browser services with some overlap. Content scripts only uses the standard web APIs, for example, HTML5, DOM, and JavaScript, with the exception of a few Chrome extension APIs. Meanwhile, a background page has full access to Chrome’s extension APIs, such as browser actions, tab events, and proxy settings, but can only access to the web APIs for its own page (a hidden web page for holding dependent web objects). Despite the API differences, content scripts and the background page can communicate via `postMessage`—the standard HTML5 message channel. Using this messaging mechanism, developers are able to coordinate content scripts and background pages for some tasks. For example, spying extensions can use content scripts to steal user information, then save the data to persistent storage in background pages.

With the necessary permissions and security context, extensions can monitor all user activities in the browser and can access sensitive user data. For example, extensions can use the cookie API to access user’s signed-in tokens, or use `webRequests` to intercept, block, or modify requests. In this study, we have found Chrome extensions that steal private user information and send the data to their remote servers.

3. DATASET

For our study, we use two types of datasets: (1) We identify and collect a sample of spying extensions from the Chrome store (`CWS_Spy`). (2) A near-complete dataset of all extensions available on the Chrome store as of September, 2016 (`CWS_All`). This dataset helps us to compare the characteristics of spying extensions with the rest of the extensions in the Chrome store.

For `CWS_Spy` and `CWS_All`, in addition to collecting the source code for each extension, we perform an additional crawl to collect all publicly available metadata associated with each extension. Metadata includes information about the size of the user base, reviews, ratings, bug reports, general functionality of the extension, and the developer.

3.1 `CWS_All` dataset

The Chrome store organizes extensions into 12 primary categories [14]. We obtain the `CWS_All` dataset by crawling all publicly visible extensions in each of the 12 categories (shown in Table 1). Our crawl conducted in September 2016, discovered 43,521 extensions (which is similar to counts reported in recent prior studies [21]).

3.2 `CWS_Spy` dataset

Obtaining a sample of spying extensions is very challenging. We rely on a combination of expert manual investigation (by the authors) and semi-automated heuristic based techniques to identify spying extensions.

We define a spying extension as one which accesses sensitive user information and sends it to remote servers (not known or trusted by the user) in a “stealthy” fashion (i.e., without any visual UI alerts or warnings), when the core functionality of the extension does not require such infor-

¹<https://developer.mozilla.org/en-US/Add-ons/WebExtensions>

Category	Chrome Store (%)	Spying (%)
Productivity	14,547 (33.42)	31 (14.22)
Fun	6,613 (15.19)	80 (36.69)
Communication	5,764 (13.24)	35 (16.05)
Web Development	4,162 (9.56)	5 (2.29)
Accessibility	4,062 (9.33)	7 (3.21)
Search Tools	2,493 (5.73)	6 (2.75)
Shopping	2,060 (4.73)	7 (3.21)
News	1,495 (3.43)	32 (14.68)
Blogging	835 (1.92)	11 (5.04)
Photos	651 (1.49)	1 (0.46)
Sports	569 (1.31)	3 (1.38)
By Google	62 (0.14)	0
Total	43,521 (CWS_A11)	218 (CWS_Spy)

Table 1: All Chrome Web Store extensions are assigned a primary category. Table shows distribution of extensions in each category and the corresponding distribution of spying extensions. Spying extensions are predominantly present in Top 3 categories by volume of extensions.

mation communication.² Table 3 shows the different types of sensitive information considered in our study. The table shows that our dataset of spying extensions belongs to diverse set of categories.

Table 1 shows high-level statistics for the collected data. We identified 218 spying extensions spanning a wide range of categories. It is interesting to note that spying extension developers target the top categories in CWS_A11 (by volume of extensions), namely, “Productivity”, “Fun”, and “Communication”, more than other categories, including “News”.

3.2.1 Verifying whether an extension is spying

We start by building a behavioral workload that includes browsing activity on the Top 10 Alexa websites (as of September 2016) [3] and use Selenium [31] for browser automation (to visit and interact with the sites in our workload).

Our high level strategy is to first identify the network events sending sensitive information to a remote server. To identify network activity triggered by an extension, we use a record-replay network proxy [29]. Starting with a fresh Chrome session, in the record phase, we run the workload without the extension loaded (*plain run*) and record all network events. Next, in the replay phase (*live run*), we run the same workload with the extension loaded, but serve requests from the cache if present. If a request misses the cache, it is routed out to the Internet. A record-replay network proxy helps to limit page dynamism and minimize inconsistencies (in network activity) between the two runs. During each run (plain and live), we record all network events, Chrome API calls, and the state of client-side storage and cookies. We use the Chrome Apps & Extensions Developer [12] to log client-side behavior (mainly Chrome API calls).

Next, we narrow down on the set of potential data stealing remote servers by identifying the network events (and their associated remote servers) present in the live run, but absent in the plain run. We inspect all the GET and POST requests made to these servers. We search for sensitive information (see Table 3 for the different types) being sent through any of these requests. We observe that sensitive information can

²We use the description provided by the extension to understand the core functionality.

be sent as plain text or in some encoded format (base64 or any custom encoding technique). In each case, we use the appropriate decoding technique if available. Once we identify an instance of information being stolen (e.g., stealing browsing history), we analyze the recorded API calls and the source code to identify the precise API calls made by the extension to access, store (if required) and send sensitive information to a remote server. In some cases, we observe that the extension requires a control trigger to move into “spying” mode (e.g., a flag set in a cookie). We try to identify such control triggers by manually inspecting the source code of the extension (e.g., by searching for code for XHR requests). We discuss more details of the different types of spying behavior in Section 4. Also, note that our methodology is not guaranteed to identify an extension as spying even if it is indeed spying (e.g., when manual source code analysis is difficult due to code obfuscation), but will never mark a benign extension as spying.

3.2.2 Identifying potential spying candidates

As our verification procedure includes manual effort, it is hard to investigate all 43,521 extensions. Instead, we use a set of heuristics to shortlist a much smaller candidate set of potentially spying extensions (out of 43k extensions) and only investigate the shortlisted set using our verification technique described in the section 3.2.1. First, we start with a set of 4 baseline techniques to build an initial candidate list. Next, we extract various signatures (discussed later in this section) from the identified spying extensions, and try to expand the initial set by searching for more extensions that match the signatures (shown in Table 2).

Baseline Techniques.

Browser extension monetization services: These are services that incentivize developers to build extensions that would be further modified to inject ads and thus help to generate some revenue for the developers (and the monetization platform receives a portion of the revenue) [33]. We found reports of a spying extension promoted by a browser monetization service [2]. Inspired by the report, we chose to identify extensions supported by such monetization platforms. Using different search queries on Google, we identify 15 browser extension monetization services³ and discover 260 extensions provided by them on the Chrome Web Store. After our verification process, we find 84 (provided by 3 out of 15 monetization services) of them to be spying extensions.

Permission based filtering : In this approach, we shortlist extensions from CWS_A11 that ask for atleast one permission known to be potentially harmful [19, 21]. This includes permissions such as *WebRequest*, and *management*. We identify 150 such extensions of which 5 are spying.

Filename based signatures: We then identify extensions that use a specific file naming scheme (for files containing code for tracking) that explicitly reveals the intention of the code. Using handcrafted regular expressions (e.g., *track*.js* or *trk*.js*), we discover 79 extensions with matching filename patterns, out of which 3 extensions are spying.

Reported extensions: In the past, there have been sev-

³Example monetization services include wips.com, addonjet.com, and addonads.com

eral reports of spying extensions on the web [23, 34]. We search the Chrome store for the reported extensions and find 8 extensions to be still alive, and verify all of them as spying extensions.

Overall, we find 100 spying extensions using the above techniques. Next, using various signatures extracted from the 100 extensions, we try to expand this set to include more spying extensions by searching for a signature match.

Spying Dataset Expansion.

Filename matching: We extract the file names of the JavaScript files which are responsible for spying behavior. We search for extensions with same file names in `CWS_A11` and find 350 more extensions out of which we verify 49 to be spying extensions.

Developer based signatures: We look for other extensions written by developers of the spying extensions in our initial list. Out of 125 such extensions, we find 8 to be spying.

URL based signatures: We extract all the URLs in each extension’s source code and search for extensions that match the URL of a known spying server. Out of 110 matches, we find 46 to be spying. For the remaining set, we suspect that these extensions are no longer tracking (or we could not trigger the spying mode). Also, some of the extensions were not functioning properly.

JavaScript based signatures: Using signatures based on code snippets responsible for spying behavior, we find 120 matching extensions. Out of these, 15 extensions are confirmed as spying. Our data expansion technique gives us an additional 118 extensions. In total, we find 218 spying extensions on the Chrome store.

Initially Found (Baseline Methods)		After Expansion	
Technique	#ext	Technique	#ext
manual inspection	5	filename matching	49
by monetization services	84	more ext by developer	8
reported extensions	8	remote URL match	46
filename based signatures	3	JS based signatures	15
Total	100		118
Total Tracking (CWS_Spy)			218

Table 2: We identify spying extensions by first inspecting an initial dataset of extensions and then expanding the dataset using the expansion techniques. We mark an extension as a spying extension only after verifying that it accesses and sends user data to a remote URL.

4. ANALYSIS OF SPYING EXTENSIONS

In this section, we analyze various aspects of spying extensions in detail.

4.1 Information Tracked

Table 3 shows the different types of sensitive information tracked by spying extensions. Once stolen, the user has no control over how this data is further used. We find that most

of the extensions spy on the browsing history of users, albeit there are two extensions only track domains that users visit. Browsing history may not only expose users to more targeted advertising, but can also lead to differential services and price discrimination on various e-commerce sites [17]. Browsing history combined with geolocation and IP address information could be used by governments to track political dissidents or by other entities interested in launching targeted attacks against the user [25]. Information about browsing history can also provide access to sensitive and private documents on services like Google Drive, Dropbox and Pastebin where a document can be accessed by anyone with link to the document. We also found spying extensions stealing Social Media access tokens and using them to access private information of users such as - photos, posts, closed groups and private messages.

Information Tracked	#Total
Browsing History	202
IP Address, Geolocation	10
OSM Access Tokens	4
Domain Visited	2
Total Unique	218

Table 3: Information Tracked by Spying Extensions. We found spying extensions to be spying on user’s browsing history, IP address, location, social media access tokens, and user visited domains.

4.2 Spying Behavior

To understand how a spying extension works, we break down the capabilities of a spying extension into following three categories: (1) ability to access specific types of sensitive user information (2) ability to store information (may not be used always) and, (3) ability to send tracked information to a remote server. Extensions require specific permissions to access various Chrome API endpoints. These special privileges can be misused by extensions to spy on users and steal personal information. In this section, we present the various permissions required for the three types of capabilities. Section 3 describes how we identify the different Chrome API endpoints required by spying extensions for accessing, storing and sending personal data. We then map the different Chrome API endpoints used by an extension to specific permission requirements [13].

4.2.1 Accessing sensitive user information

Table 4 shows a list of permission requirements for accessing sensitive user information by spying extensions. For comparison purposes, we also show the number (and percentage) of all other extensions (`CWS_A11 \ CWS_Spy`) that use the same permissions (possibly for other purposes). Spying extensions can continuously monitor user’s browsing behavior with the help of ‘tabs’, ‘activeTab’ and ‘all urls’ permissions which enable them to access the URL being visited at the moment by a user. We observe that the ‘tabs’ permission ranks at the top (being used by over 94% of spying extensions), but also used by a majority of all other extensions. Similarly, the ‘cookies’ permission is also used by most of the spying extensions (e.g., to access social media access tokens), but only by a smaller fraction (8.9%) of all other extensions. Browsing history and location can also be tracked by direct

access to ‘history’ and ‘geolocation’ permissions.

Permission	CWS_Spy #ext (%)	CWS_A11 \ CWS_Spy #ext (%)
1 tabs	207 (94.95)	22483 (52.24)
2 cookies	181 (83.03)	3844 (8.93)
3 storage	22 (10.09)	13376 (31.08)
4 all urls	14 (6.42)	4022 (9.35)
5 history	6 (2.75)	862 (2.0)
6 geolocation	3 (1.38)	378 (0.88)
7 activeTab	3 (1.38)	5920 (13.76)

Table 4: Permissions required by spying extensions to access sensitive user information.

4.2.2 Storing sensitive user information

Spying extensions may or may not store user information on client side. If they do, the client-side cookies and ‘unlimited storage’ are used to store information either in plain text or in obfuscated form. This data is stored and accessed by extensions at regular intervals or periodically by a CnC remote server controlling the extension. We find that most of the extensions (188 out of 218) are storing sensitive information before sending it to remote servers. Thus inspecting local storage would be useful in identifying spying behavior. Table 5 lists the permissions to store user information and their distributions used by tracking and other extensions.

Permission	CWS_Spy #ext (%)	CWS_A11 \ CWS_Spy #ext (%)
1 cookies	181 (83.03)	3844 (8.93)
2 unlimitedStorage	16 (7.34)	2737 (6.32)

Table 5: Permissions required by spying extensions to store information on the client-side.

4.2.3 Sending sensitive user information

Spying extensions use various techniques to send user information to remote servers. One of the methods used is to pass user information (either as it is, or obfuscated) as parameters of a GET request to fetch resources (e.g., an image) from the remote server. Extensions also use XMLHttpRequest permission provided by browser to observe and analyze traffic and to intercept, block, or modify requests in-flight. This permission lets the extensions make XMLHttpRequests to an untrusted remote server and is the most common method for sending sensitive information. Table 6 shows distribution of spying extensions using XMLHttpRequest permission to send user data.

Permission	CWS_Spy #ext (%)	CWS_A11 \ CWS_Spy #ext (%)
1 XMLHttpRequest	140 (64.22)	4784 (11.05)

Table 6: Permissions required by chrome extensions to send client information to a remote server.

4.2.4 Control of spying extension by a CnC server

We observe that some extensions do not start spying on users at the time of installation. They need to be triggered by a remote server to either start collecting user data, or

send the stored user data to the remote server. We observe two methods being used to control spying state: (1) toggling flags in cookies and client side storage, and (2) time based triggers which send information after a certain period of time. This functionality makes detection of spying extensions a challenging task since they may act as benign during some period of time. We identify 88 extensions where the tracking code was toggled on and off using a state variable in the cookie store using ‘cookie.set’ API call. We find 2 extensions using time based triggers which starts tracking after fixed periods according to a time defined in a source JavaScript file. These control triggers requires the ‘webRequest’ and ‘cookies’ permissions. We were unable to identify any control triggers in the other spying extensions.

4.3 Remote spying entity

In this section, we investigate the remote spying entities (or trackers) associated with each extension. We find that the 218 spying extensions sent sensitive user information to 29 unique tracker domains⁴. We notice that 6 spying extensions were contacting more than one tracker. Interestingly, 5 out of 29 tracker domains belong to known browser extension monetization services (see earlier Section 3). It is unclear how this data is further used by the monetization service, but could probably be sold to other parties for profit or used for highly targeted behavioral advertising.

To understand the reputation of tracker domains, we query various popular blacklisting services like - Google safeBrowsing API, VirusTotal and Phishtank. We also analyze their WoT ranking which is a crowd-sourced website reputation metric [1]. None of the trackers were blacklisted by any of the services, indicating that these tracker domains are difficult to detect. In fact, 8 tracker domains have a WOT score of over 9.0 which indicates very positive crowdsourced feedback.

4.4 User Base

In this section, we analyze the reach of spying extensions in terms of number of installs from the Chrome store. Figure 1 shows the distribution of number of users of spying extensions and all extensions on the Chrome store. It can be seen that there is no significant difference between the two. All spying extensions have at least one user, and around 30% of them have more than 1,000 users. In comparison, there are 0.43% extensions with no users on the Chrome store. We find that spying extensions have over 2.4 million cumulative installs. Even though we expect the fraction of spying extensions in the Chrome store to be very small (prior work found 0.27% of extensions in the Chrome store to be malicious [21] and spying is a subset of malicious extensions), a significant fraction of spying extensions are widely installed and thus it is important to identify and take down spying extensions from the Chrome store.

4.5 Crowd-sourced reputation / feedback of spying extensions

In this section, we first analyze the ratings received by spying extensions. Next, we investigate whether they receive any complaints, bug reports or questions. This would give us a sense for the awareness of Chrome users about the spying behavior of these extensions.

⁴Tracker domain is the untrusted remote server to where the user information is being sent

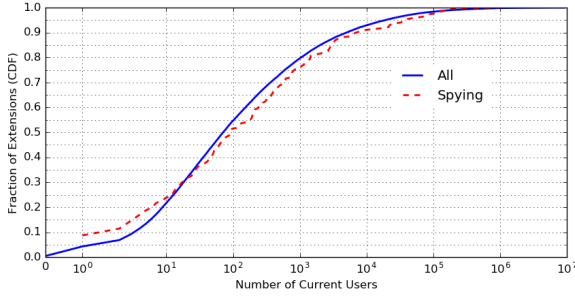


Figure 1: Number of current users for all the Chrome store extensions

4.5.1 Ratings

Figure 2 shows that spying extensions receive mostly similar ratings as other extensions on the Chrome store. Spying extensions have a median rating of 4.1, while the median rating for all extensions is 4.4. 50.91% of spying extensions receive a rating higher than 4.0. Even the distribution of number of ratings received by spying extensions is similar to that received by all extensions on the Chrome store (Figure 3). Thus, it would be hard to distinguish between spying and benign extensions based on crowd-sourced feedback for both Google and for users trying to decide whether to install an extension.

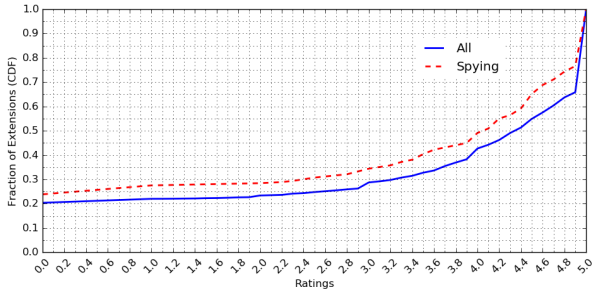


Figure 2: Crowdsourced average rating of extensions on Chrome Store

4.5.2 Bug Reports, Questions and Complaints

Chrome Web Store lets users submit questions or post suggestions and problems pertaining to extensions. We crawl this textual data and analyze it to identify if users are reporting bugs or are complaining about the functionality of the extension. Out of 218 extensions, 103 extensions received at least one comment by a user. We scan the text reports using a set of keywords (based on manual investigation) indicating suspicious behavior (e.g., ‘tracking’, ‘keylogger’, ‘steal’, ‘fake’, ‘dont install’, ‘malware’). Interestingly, only 12 (5.5%) extensions received some form of comment reporting suspicious behavior. Our findings suggest that users are potentially unaware of the spying behavior of these extensions.

4.6 Developers

There are 69 unique developers for our dataset of 218 spying extensions. A large fraction of developers, 89% develop

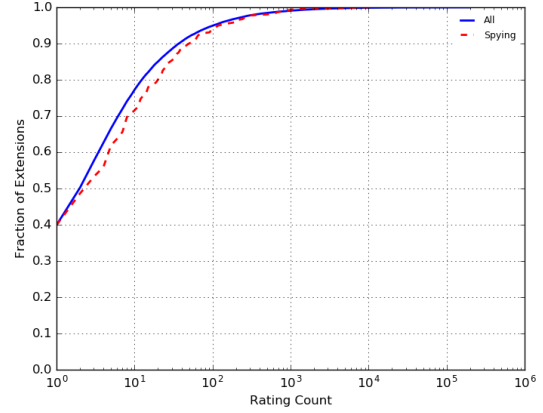


Figure 3: Number of votes (count) contributing rating of extensions on Chrome store

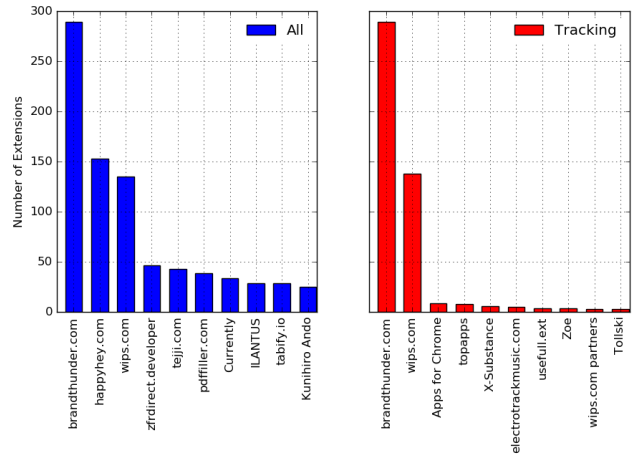


Figure 4: Top 10 developers (by volume of extensions) on Chrome store

only a single spying extension. This trend is not surprising, as Google might target and investigate other extensions by the same developer if one malicious extension is caught. However, we observe that a small fraction, 5 spying extension developers also show up among the top 500 out of the total 33,477 developers in the entire Chrome store (based on number of cumulative installs). We also look at the developers who publish maximum number of extensions on the Chrome Web Store. Figure 4 shows that 2 of the top 10 developers (by volume of extensions) have published spying extensions. This indicates that even popular developers can have a few spying extensions which can go undetected and extensions from popular developers can not be trusted to be safe.

5. DETECTING SPYING EXTENSIONS

Our methodology for identifying spying extensions, described in Section 3.2 involves manual effort. Now that we have identified a reasonably large sample of spying exten-

χ^2	Feature	Cat.
11462.8	[cookies.set, webRequest.onBeforeSendHeaders]	F2
8119.8	[cookies, management, notifications]	F1
7649.3	[management, tabs]	F1
6873.8	[dom.access, tabs.get]	F2
6641.1	localStorage	F4
6327.8	[all_urls, tab]	F1
5382.9	WOT Ranking of remote URLs	F5
4091.1	XML HTTP Request	F5
4031.9	number of Eval functions	F3
3773.3	[contextMenus, unlimitedStorage]	F1

Table 8: χ^2 Ranked list of features to detect tracking extensions for each category (Cat.) of features.

sions, we investigate whether we can automatically detect spying extensions using machine learning schemes.

5.1 Feature Extraction

We begin by extracting an exhaustive set of features to detect spying extensions. We run every extension in a controlled environment and follow the methodology explained in Section 3.2.1 to record client-side and network behavior. All the features are described in detail in Table 7. We extract these features for all 43,521 extensions (including spying extensions) and prepare a total of 42,368 features. Note that our feature set includes most of the features (whenever available) used in prior work on detecting malicious browser extensions [21].

5.2 Feature Ranking

Table 8 shows the ranked list of features and their corresponding feature categories based on the χ^2 test. In the ranked list, 4 out of the top 5 features fall in categories F1 and F2 and reflect the permission requirements and Chrome API calls required for spying behavior. Examples include permission to access ‘tabs’ and ‘cookies’, as well as key Chrome API calls required to send data (e.g., webRequest). These are robust features because they constitute the core permission requirements and Chrome API calls required for stealing sensitive information, and would thus make it hard for an extension to evade detection.

5.3 Detection performance

We now use the previously described features to detect spying extensions using machine learning. We train various classifier models using our labeled dataset of 218 tracking extensions (as positive class) and 43,303 other Chrome Web Store extensions (as negative class). We assume that a significant majority of extensions in the negative class are not spying (prior work estimated fraction of malicious extensions to be around 0.27% [21]). Note that our training set is highly imbalanced and likely reflects the true imbalance ratio an operator might encounter when deploying a classifier for identifying spying extensions in the Chrome store. High imbalance in the training set makes our classification problem even more challenging.

We use multiple classifiers including Decision Trees, Random Forest, Adaboost (an ensemble method using Random Forest as the base estimator), SVM (with a RBF kernel) and a Neural Network. For the neural network, we use a network architecture with a single hidden layer having 100 neurons and use stochastic gradient descent with a ‘reLU’ activation

function. We also use L2 regularization with a high penalty value, $\alpha = 1.0$ to limit overfitting. We use 70:30 training-testing split of each class in our dataset and perform 5-fold cross validation. Precision and recall values for the spying class are reported in Table 9.

Using all features, we achieve the highest precision and recall using the neural network. It is interesting to note that the classification performance difference between the neural network classifier and others is huge. To further investigate whether the neural network was overfitting, we calculate precision and recall of training and validation set over multiple iterations of the training phase. We observe in Figure 5 that training and validation set have very similar curves for precision and recall, indicating a good quality classifier (with limited overfitting).

Also, among the different feature categories, we obtain the highest classification accuracy with category F2 (features based on Chrome API calls). This is a promising result as these features are harder to manipulate, as some of them include the core API calls required for stealing sensitive information.

Thus it is possible to detect a large number (over 53% recall) of spying extensions with high accuracy (86% precision). This means that there is significant scope for automatically detecting spying extensions in the Chrome store with further improvements to be made by using more advanced neural network architectures.

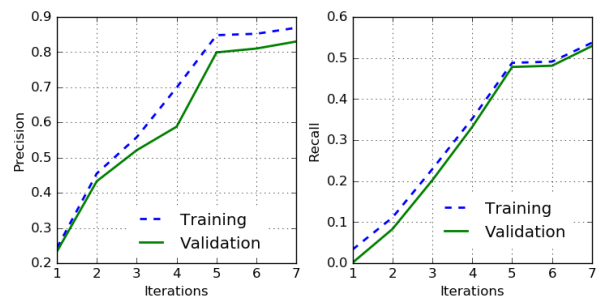


Figure 5: Precision and Recall using a Neural Network over different iterations.

6. RELATED WORK

There are two main directions of related work:

User Behavior Tracking: There have been several evidences and studies of user behavior tracking across the Web [9, 24, 27]. These studies show the evidence of web trackers present on websites which can have access to various sensitive private information of the user like browsing history, user name, OSN identifiers [22], cookies [10] and even predict user’s shopping trends [16]. Studies show that ad-injection over the web can help monitor user’s browsing behavior [33] and user engagement [7]. There have been active efforts to protect users against web tracking by providing various browser solutions to identify and block third-party trackers [26]. Web based trackers however provide a fragmented view of user behavior. In this paper, we study tracking via browser extensions which can track users across the web.

Analysis and Detection of Malicious Browser Extensions: Rogue browser extensions have been on the

Feature Set		Feature Description
F1	Permission	<i>Permissions</i> : Binary vector, each value denoting presence or absence of a permission such as local-storage, tabs, cookies <i>Permission Combination</i> : Binary vector capturing the presence or absence of bigrams and trigrams of permission types
F2	Chrome API Calls	<i>API Call</i> : Binary vector, each value denoting a call to specific Chrome API <i>API Call Sequence</i> : Binary vector capturing the presence or absence of bigrams and trigrams of Chrome API call sequences
F3	JavaScript Based	<i>Eval</i> : Binary value to show the presence of eval function used by an extension in its code <i>base64</i> : Binary value to show the presence of extension using base64 encoding of data
F4	Client Side Storage	<i>Cookies</i> : Binary value to capture state change of cookies after loading the extensions <i>Storage</i> : Binary value to capture state change of local client-side storage after loading extension <i>URL in Cookies</i> : Binary value to capture if a URL was stored in cookies <i>URL in Storage</i> : Binary value to capture if a URL was stored in the client-side storage
F5	Network Log	<i>XML HTTP</i> : Number of XHR calls made by the extension <i>GET</i> : Number of GET queries invoked by the extension <i>POST</i> : Number of POST queries by the extension <i>Remote URL</i> : Binary value denoting whether a network request is present in live run that is not present in the plain run <i>WoT</i> : Crowd-sourced reputation of a domain contacted in the live run, but not contacted in the plain run (if any)
F6	Others	<i>Filename Match</i> : Binary value denoting whether extension contains files with suspicious filenames (based on a pre-defined list discussed in Section 3.2) <i>Metadata</i> : Rating, number of reviews, number of users of every extension

Table 7: Feature description to perform automated detection of spying extensions.

Features	DecisionTree		Random Forest		SVM (rbf)		Adaboost		Neural network	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
F1	5.67	7.9	9.87	12.82	8.22	12.23	10.11	13.21	59.33	39.1
F2	15.55	10.12	20.82	23.11	20.23	23.23	24.23	23.24	82.82	53.56
F3	15.45	12.67	19.76	13.12	12.22	31.23	14.56	29.98	60.21	48.34
F4	13.55	15.03	13.66	17.91	15.54	20.12	16.12	29.91	59.91	35.20
F5	15.09	12.90	20.02	19.98	13.62	30.32	21.23	25.21	61.21	52.10
F6	17.45	15.01	19.88	15.02	9.87	12.21	20.22	15.23	62.91	49.21
All Feats	20.23	20.01	21.29	24.12	26.23	24.15	28.23	27.82	86.91	53.74

Table 9: Precision and recall (in %) of tracking extensions using machine learning classification

rise to spread malware and use user tracking and web traffic to generate revenues [19]. Studies have uncovered rogue behavior of extensions such as injecting ads, spreading malware, tracking and OSN account hijacking [19, 21]. To understand how rogue extensions operate, research has also explored the permission model of extensions and the associated meta information [5, 11]. There have also been other techniques used to detect malicious code, specially which change the DOM elements of webpages by leveraging event based fuzzers and using the permission and metadata to detect malicious extensions [19, 21, 32]. Extension’s source code can give hint about possible malicious behavior. However, extension’s source code can be complex and challenging for performing static code analysis due to the presence of obfuscated and minified code. Apart from using static code analysis, researchers have performed information flow analysis of browser extensions and JavaScript based applications to identify security vulnerabilities and track the flow of sensitive information because of malicious functionality of browser extensions [4, 8, 20]. In this work we focus on the permissions model and browser API calls available to extensions to identify spying extensions.

7. CONCLUSION

In this work, we conduct the first large-scale study of spy-

ing browser extensions. We identify and analyze extensions stealing a variety of sensitive personal information, including the browsing history of the user. In fact, being a privileged software residing in the user’s browser, spying extensions can track the complete online activity of a user. Unfortunately, these spying extensions are able to attract a huge user base. Our dataset of 218 spying extensions had over 2.4M cumulative installs. Moreover, at the time of this writing, only 4 out of 218 extensions have been removed from the Chrome store. While prior work on user tracking has largely focused on third-party tracking on the web, our work highlights the importance of detecting and limiting user tracking within the browser. We had an interaction with the Google Chrome Security Team regarding this research work.

8. REFERENCES

- [1] Web of trust (wot). <https://www.mywot.com/>.
- [2] Chrome security flaw allows android apps to be installed without user knowledge. <http://gadgets.ndtv.com/apps/news/chrome-security-flaw-allows-android-apps-to-be-installed-without-user-knowledge-625255>, November 2014.
- [3] Alexa. Actionable analytics for web. <http://www.alexa.com/>.

- [4] S. Bandhakavi, N. Tiku, W. Pittman, S. T. King, P. Madhusudan, and M. Winslett. Vetting browser extensions for security vulnerabilities with vex. *Communications of the ACM*, 54(9):91–99, 2011.
- [5] L. Bauer, S. Cai, L. Jia, T. Passaro, and Y. Tian. Analyzing the dangers posed by chrome extensions. In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pages 184–192. IEEE, 2014.
- [6] I. A. Bureau. First quarter u.s. internet ad revenues hit record-setting high at nearly 16 billion usd, June 2016.
- [7] V. Dave, S. Guha, and Y. Zhang. Measuring and fingerprinting click-spam in ad networks. *ACM SIGCOMM Computer Communication Review*, 42(4):175–186, 2012.
- [8] M. Dhawan and V. Ganapathy. Analyzing information flow in javascript-based browser extensions. In *Computer Security Applications Conference, 2009. ACSAC’09. Annual*, pages 382–391. IEEE, 2009.
- [9] S. Englehardt and A. Narayanan. Online tracking: A 1-million-site measurement and analysis. Technical report, Technical report, May, 2016.
- [10] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten. Cookies that give you away: The surveillance implications of web tracking. In *24th International Conference on World Wide Web*, 2015.
- [11] A. P. Felt, K. Greenwood, and D. Wagner. The effectiveness of application permissions. In *Proceedings of the 2nd USENIX conference on Web application development*, pages 7–7, 2011.
- [12] Google. Chrome apps and extensions developer tool. <https://chrome.google.com/webstore/detail/chrome-apps-extensions-de/ohmmkhmmmpcnpikjeljgnaobkaalbgc>.
- [13] Google. Chrome permissions. developer.chrome.com/extensions/declare_permissions.
- [14] Google. Chrome web store. <https://chrome.google.com/webstore/category/extensions>.
- [15] Grammarly. Automated english language proofreader. <https://www.grammarly.com>.
- [16] Q. Guo and E. Agichtein. Ready to buy or just browsing?: detecting web searcher goals from interaction data. In *Proceedings of the 33rd international ACM SIGIR conference*, pages 130–137. ACM, 2010.
- [17] A. Hannak, G. Soeller, D. Lazer, A. Mislove, and C. Wilson. Measuring price discrimination and steering on e-commerce web sites. In *Proceedings of the 2014 conference on internet measurement conference*, pages 305–318. ACM, 2014.
- [18] Honey. Automatically find and apply coupon codes. <https://www.joinhoney.com/install>.
- [19] N. Jagpal, E. Dingle, J.-P. Gravel, P. Mavrommatis, N. Provos, M. A. Rajab, and K. Thomas. Trends and lessons from three years fighting malicious extensions. In *24th USENIX Security Symposium*, 2015.
- [20] S. Just, A. Cleary, B. Shirley, and C. Hammer. Information flow analysis for javascript. In *Proceedings of the 1st ACM SIGPLAN international workshop on Programming language and systems technologies for internet clients*, pages 9–18. ACM, 2011.
- [21] A. Kapravelos, C. Grier, N. Chachra, C. Kruegel, G. Vigna, and V. Paxson. Hulk: Eliciting malicious behavior in browser extensions. In *23rd USENIX Security Symposium (USENIX Security 14)*, pages 641–654, 2014.
- [22] B. Krishnamurthy and C. E. Wills. On the leakage of personally identifiable information via online social networks. In *Proceedings of the 2nd ACM workshop on Online social networks*, pages 7–12. ACM, 2009.
- [23] D. Labs. Chrome extensions - aka total absence of privacy. <https://labs.detectify.com/2015/11/19/chrome-extensions-aka-total-absence-of-privacy/>, November 2015.
- [24] A. Lerner, A. K. Simpson, T. Kohno, and F. Roesner. Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016. In *25th USENIX Security Symposium (USENIX Security 16)*, 2016.
- [25] W. R. Marczak, J. Scott-Railton, M. Marquis-Boire, and V. Paxson. When governments hack opponents: A look at actors and technology. In *23rd USENIX Security Symposium*, 2014.
- [26] J. Mayer and A. Narayanan. Do not track. <http://donottrack.us/>.
- [27] J. R. Mayer and J. C. Mitchell. Third-party web tracking: Policy and technology. In *2012 IEEE Symposium on Security and Privacy*, pages 413–427. IEEE, 2012.
- [28] C. Reis. *Web browsers as operating systems: supporting robust and secure web programs*. PhD thesis, University of Washington, 2009.
- [29] W. P. Replay. <https://github.com/chromium/web-page-replay>.
- [30] F. Roesner, T. Kohno, and D. Wetherall. Detecting and defending against third-party tracking on the web. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 12–12. USENIX Association, 2012.
- [31] Selenium. Browser automation. <http://docs.seleniumhq.org/>.
- [32] H. Shahriar, K. Weldemariam, M. Zulkernine, and T. Lutellier. Effective detection of vulnerable and malicious browser extensions. *Computers & Security*, 47:66–84, 2014.
- [33] K. Thomas, E. Bursztein, C. Grier, G. Ho, N. Jagpal, A. Kapravelos, D. McCoy, A. Nappa, V. Paxson, P. Pearce, et al. Ad injection at scale: Assessing deceptive advertisement modifications. In *2015 IEEE Symposium on Security and Privacy*.
- [34] M. Weissbacher. These chrome extensions spy on 8 million users. <http://mweissbacher.com/blog/2016/03/31/these-chrome-extensions-spy-on-8-million-users/>, March 2016.
- [35] X. Xing, W. Meng, B. Lee, U. Weinsberg, A. Sheth, R. Perdisci, and W. Lee. Understanding malvertising through ad-injecting browser extensions. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1286–1295. ACM, 2015.